Data Analysis and Visualization in R (IN2339)

Exercise Session 9 - Statistical Assessments for Big Data

Jun Cheng, Christian Mertes, Vicente Yépez, Julien Gagneur

Section 00 - Getting ready

```
library(ggplot2)
library(data.table)
library(magrittr)
library(tidyr)
library(dplyr)
library(patchwork)
```

Section 01 - Quantile-Quantile plots

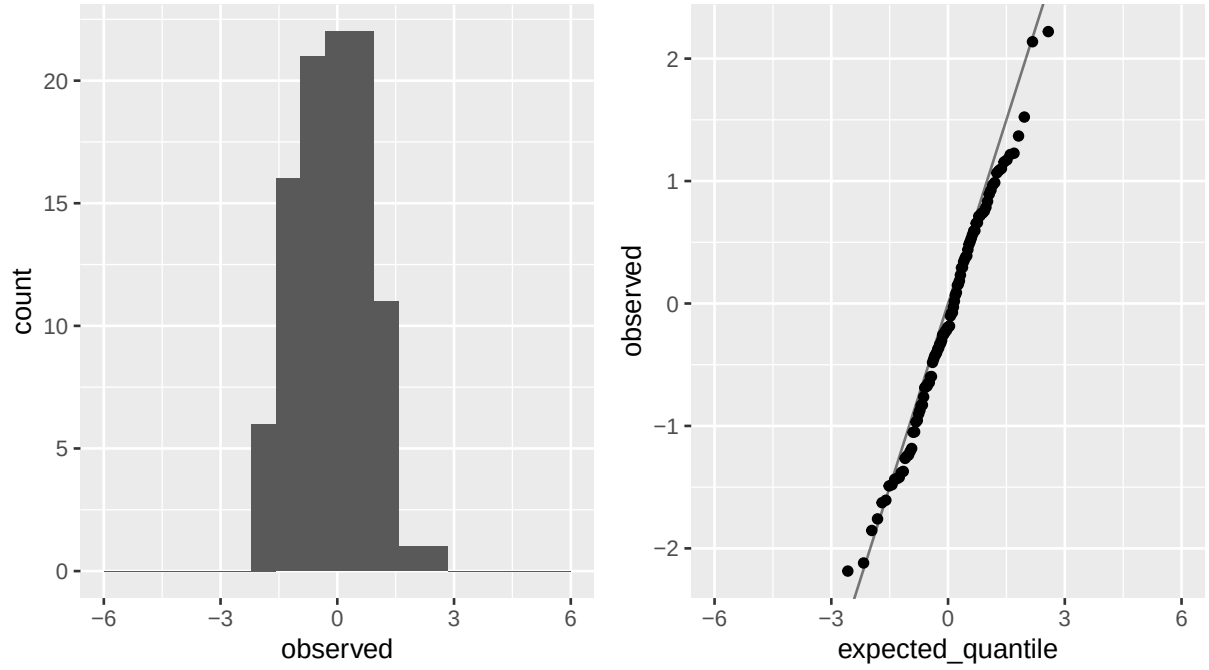
We will simulate some data from different distributions and will compare their quantile-quantile plots.

1. We will use a standard normal ($\mu = 0$ and $\sigma^2 = 1$) distribution as a reference set. Please create a data table with 100 observations with a column containing the simulated observed values. Plot first a histogram of the observed values. Then create a column containing the expected quantiles and plot the expected against the observed quantiles.

Hint: Set the x and y limits to `[-6,6]` where appropriate.

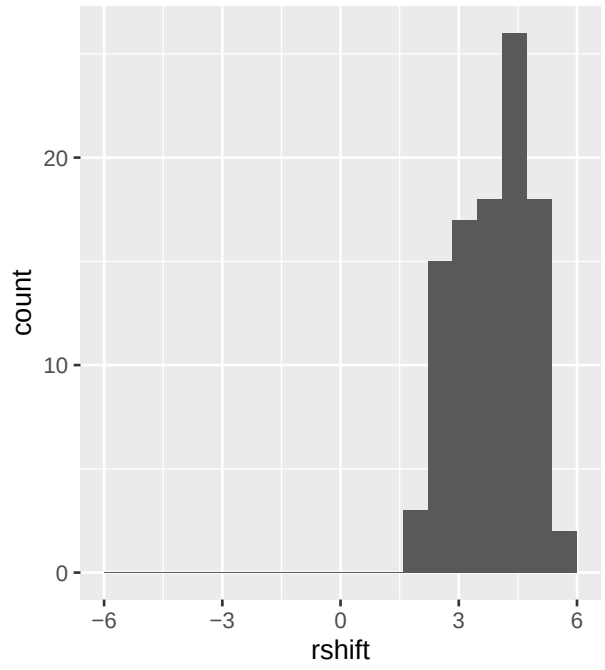
```
set.seed(10)

plot_qq <- function(dt, observed_quantiles){

  histo <- ggplot(dt, aes(get(observed_quantiles))) + geom_histogram(bins=20) +
    xlim(-6,6) +
    labs(x = observed_quantiles)

  qq <- ggplot(dt) +
    geom_point(aes(expected_quantile, get(observed_quantiles))) +
    geom_abline(intercept = 0, slope = 1, alpha = 0.5) +
    xlim(-6,6) +
    labs(y = observed_quantiles)

  histo + qq

}

# qnorm returns the quantiles of a normal distribution and ppoints returns a uniform distribution
# with n points.
```

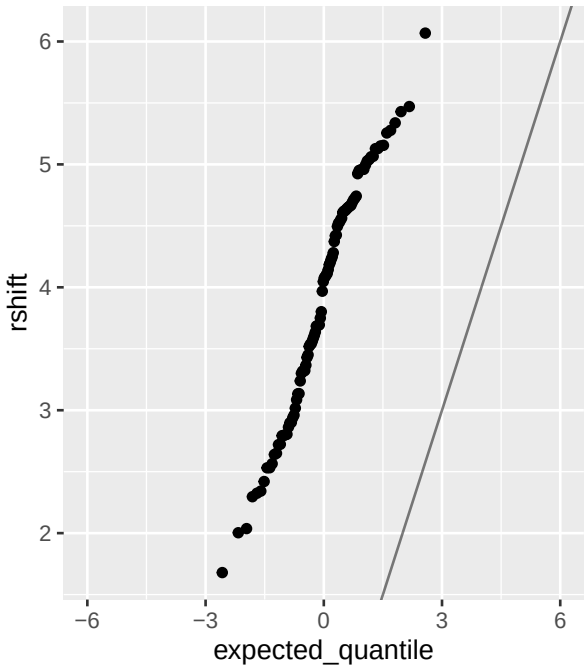
```r
n <- 100
dt <- data.table(
  expected_quantile = qnorm(ppoints(n)),
  observed = sort(rnorm(n)))

plot_qq(dt, "observed")
```



2. Now add a normal distribution with $\mu = 4$ to your `data.table` and plot the Q-Q plot. How did it change?

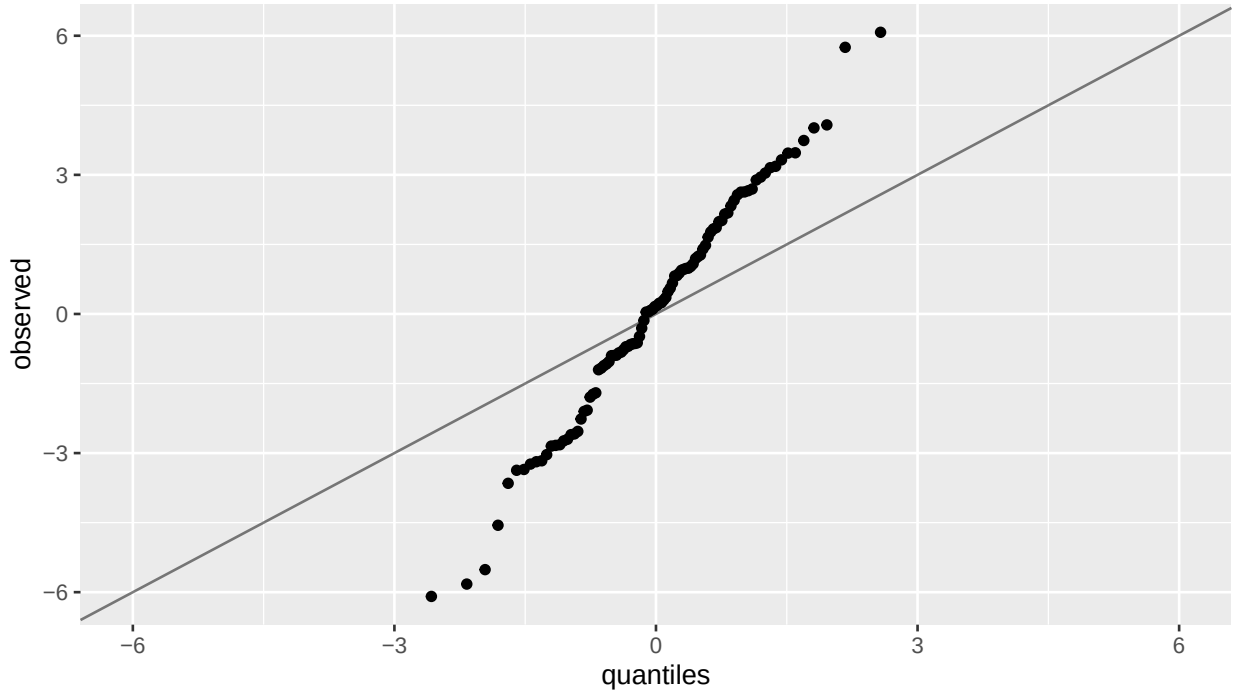
```r
dt[, rshift := sort(rnorm(n, mean=4))]
plot_qq(dt, "rshift")
```





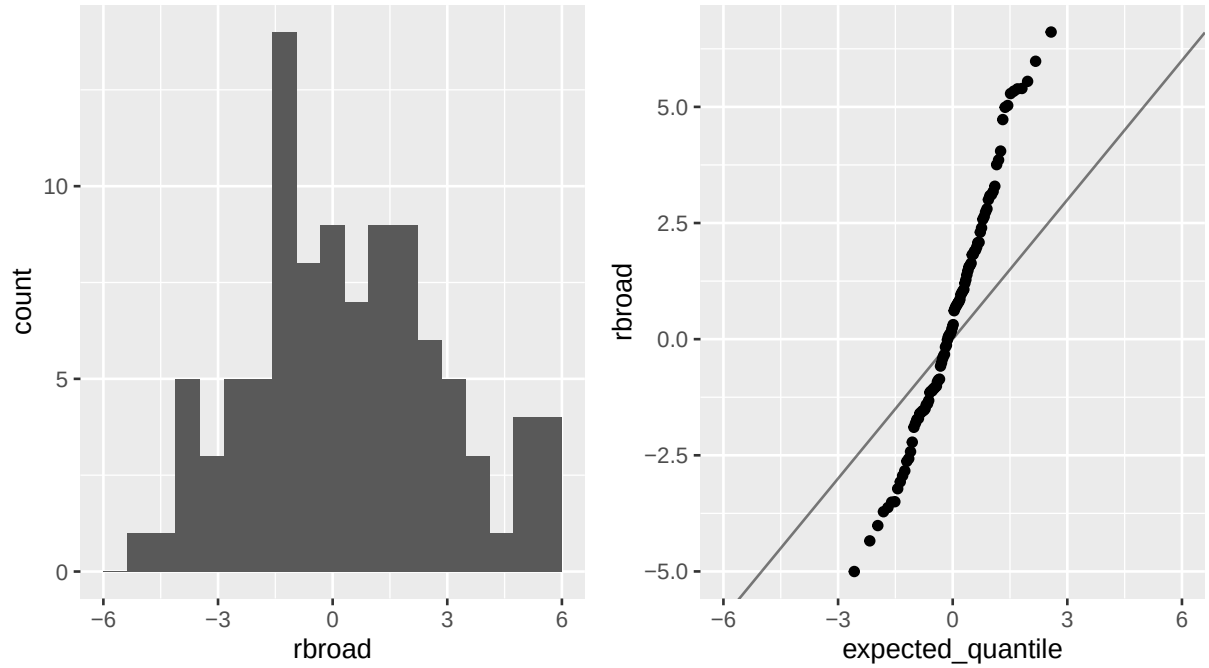

```
## The shift in the mean of the distribution is in relation to the shift from the diagonal.
```

3. How would you tweak the distribution so that you get the following Q-Q plot?




```
# Above we can observe, that the observed values cover a larger range than expected.
# Changing the standard deviation of the distribution changes the slope of the Q-Q plot.
# You can find the correct standard deviation by experimenting with values larger than 1.
```

```
dt[, rbroad := sort(rnorm(n, sd=2.5))]
plot_qq(dt, "rbroad")
```



Section 02 - QTL mapping of growth

For the next questions, we will use the yeast dataset.

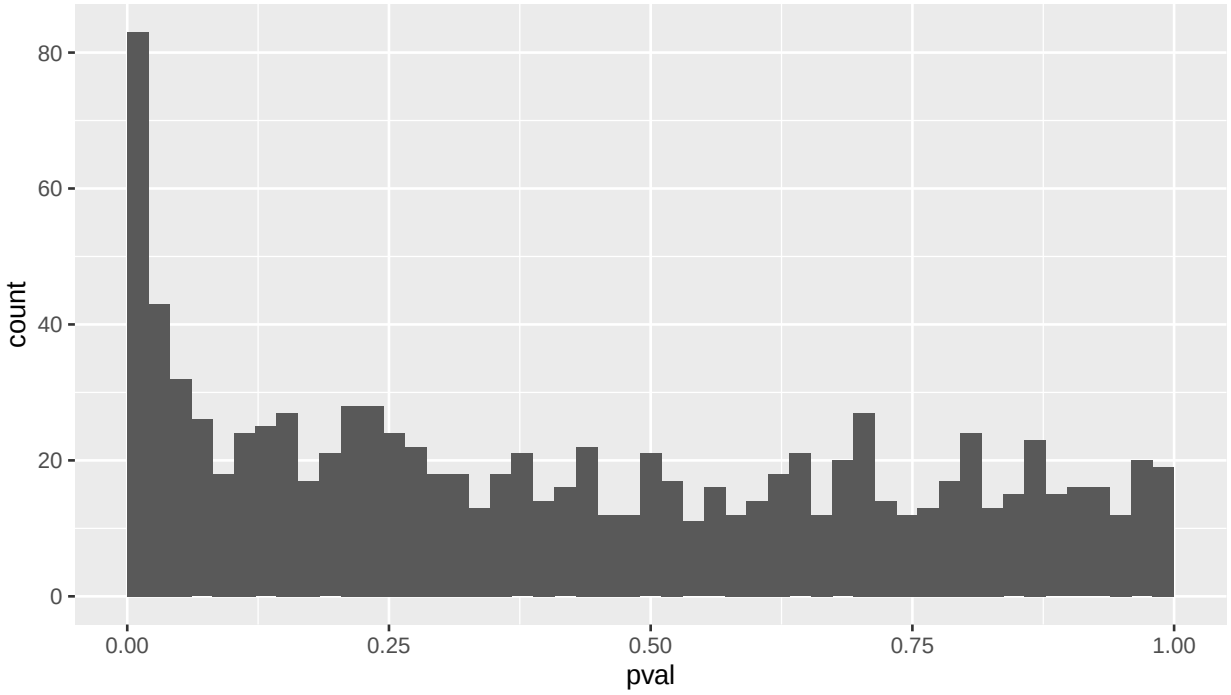
```
library(tidyr)
library(data.table)
library(ggplot2)
library(ggthemes)

## load the data
genotype <- fread("extdata/eqtl/genotype.txt")
growth_rate <- fread("extdata/eqtl/growth.txt")
marker <- fread("extdata/eqtl/marker.txt")
setnames(marker, 'id', 'marker')
genotype <- genotype %>%
  melt(id.vars = 'strain', variable.name = 'marker', value.name = 'genotype')
```

1. Test for markers associated with growth.

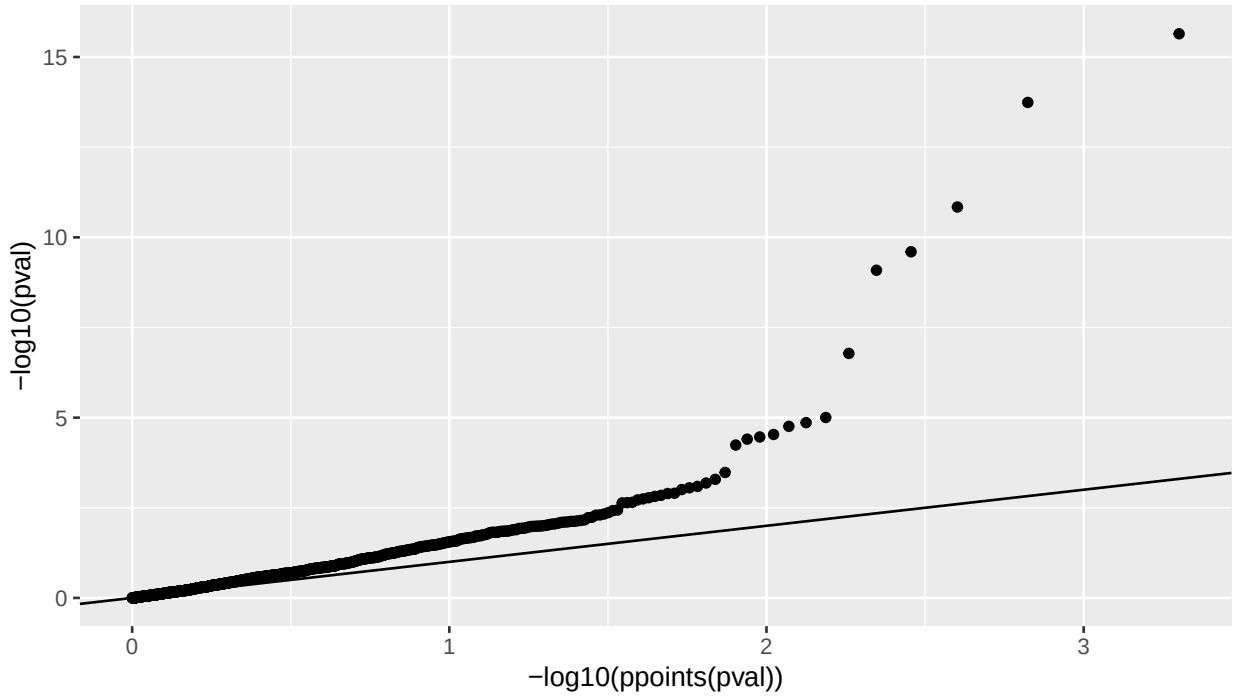
Look for a genetic marker associating with growth rate in maltose. To do so, run a wilcoxon test for growth rate versus the genotype at each of the 1,000 markers. Remember that we tested this relationship for one specific marker last time. Plot a histogram and a Q-Q plot of the obtained p-values. Which ones would you consider significant and why? Do we need to correct for multiple testing?

Hint: plot p-values in `-log10` scale.

```r
# we first need to merge the genotype and the growth rate tables.
genotype_growth <- merge(genotype, growth_rate, by = 'strain')

# We can then execute the test for each marker inside a data.table function and extract the p-values.
test_res <- genotype_growth[, .(pval=wilcox.test(YPMalt ~ genotype)$p.value), by='marker']


ggplot(test_res, aes(pval)) + geom_histogram(boundary = TRUE, bins=50)
```




```r
ggplot(test_res[order(pval)], aes(-log10(ppoints(pval)), -log10(pval))) + geom_point() + geom_abline()
```




```
# We need to correct for multiple testing, as we just made 1000 tests.
# To do so we use the Benjamini-Hochberg method implemented in the p.adjust function.
test_res[, padj:=p.adjust(pval, method='BH')]
test_res[padj < 0.1][order(padj)]
```

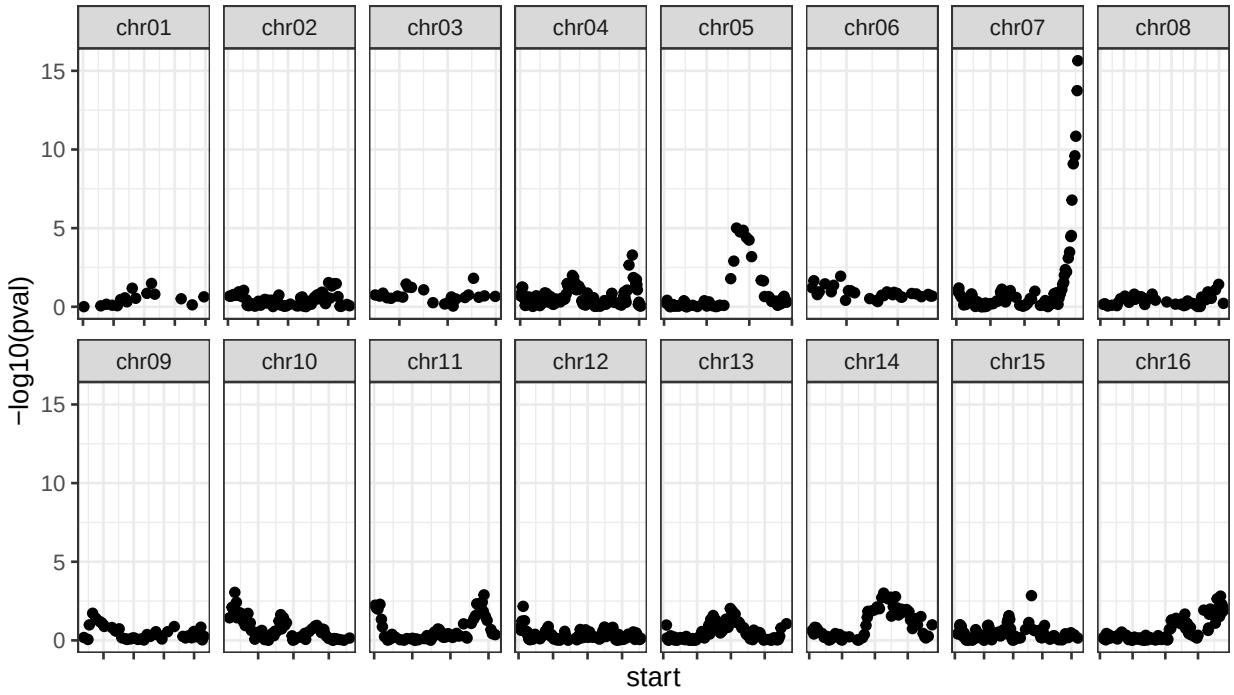
```
##        marker         pval         padj
##  1:  mrk_5211 2.264386e-16 2.264386e-13
##  2:  mrk_5198 1.813867e-14 9.069334e-12
##  3:  mrk_5184 1.440971e-11 4.803237e-09
##  4:  mrk_5171 2.518482e-10 6.296204e-08
##  5:  mrk_5158 8.174932e-10 1.634986e-07
##  6:  mrk_5144 1.657134e-07 2.761891e-05
##  7:  mrk_3359 9.918284e-06 1.416898e-03
##  8:  mrk_3385 1.376004e-05 1.720005e-03
##  9:  mrk_3372 1.747656e-05 1.941840e-03
## 10:  mrk_5131 2.932144e-05 2.932144e-03
## 11:  mrk_5118 3.435018e-05 3.122744e-03
## 12:  mrk_3399 3.954888e-05 3.295740e-03
## 13:  mrk_3412 5.755220e-05 4.427092e-03
## 14:  mrk_5104 3.317282e-04 2.369487e-02
## 15:  mrk_2799 5.154019e-04 3.436012e-02
## 16:  mrk_3425 6.502705e-04 4.064191e-02
## 17:  mrk_5091 8.088140e-04 4.757730e-02
## 18:  mrk_6384 8.831007e-04 4.906115e-02
## 19: mrk_10688 9.876249e-04 5.198026e-02
## 20:  mrk_3345 1.252817e-03 6.056720e-02
## 21:  mrk_7876 1.271911e-03 6.056720e-02
## 22: mrk_11914 1.430328e-03 6.501493e-02
## 23: mrk_13247 1.529206e-03 6.648722e-02
## 24: mrk_10755 1.661724e-03 6.923848e-02
```

```
## 25: mrk_10715 1.792178e-03 7.168713e-02
## 26: mrk_10675 1.908980e-03 7.342229e-02
## 27:  mrk_2786 2.235172e-03 7.977454e-02
## 28: mrk_10702 2.313462e-03 7.977454e-02
## 29: mrk_13207 2.273977e-03 7.977454e-02
##        marker         pval         padj
```

2. Plot the p-values against genomic position. Do you see positions that associated with growth? The genomic position is defined by the chromosome the marker is on and the marker's position within that chromosome.

Hint: plot p-values in `-log10` scale. Use `start` column for position from the marker table. In ggplot, use facet on chromosome.

```
# get marker position
marker_pval <- merge(marker, test_res, by = "marker")
ggplot(marker_pval, aes(start, -log10(pval))) +
  geom_point() +
  facet_wrap(~chrom, scales = 'free_x', nrow = 2) +
  theme_bw() +
  theme(axis.text.x = element_blank())
```



3. How many marker significantly associate with growth after correcting for multiple testing? Find the two markers with the lowest p-values.

```
marker_pval[padj < 0.05, .N]
```

```
## [1] 18
```

```
head(marker_pval[padj < 0.05][order(pval)])
```

```
##      marker chrom   start     end         pval         padj
## 1: mrk_5211 chr07 1069229 1069336 2.264386e-16 2.264386e-13
## 2: mrk_5198 chr07 1063663 1063663 1.813867e-14 9.069334e-12
## 3: mrk_5184 chr07 1051752 1051752 1.440971e-11 4.803237e-09
## 4: mrk_5171 chr07 1043492 1043503 2.518482e-10 6.296204e-08
## 5: mrk_5158 chr07 1029719 1032044 8.174932e-10 1.634986e-07
## 6: mrk_5144 chr07 1017913 1017913 1.657134e-07 2.761891e-05
```

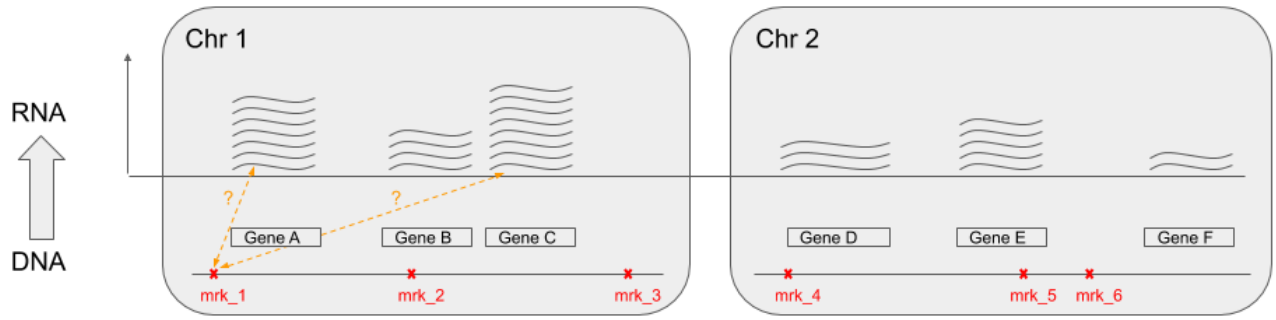
Section 03 - QTL mapping of gene expression (eQTL)

In this exercise, we move to a much larger dataset. RNA abundances have been measured for 8,382 genes in different strains of yeast grown in different environments.

Gene expression

The overall fitness, for example the growth rate of a micro-organism, is the outcome of complicated molecular processes occurring in cells. One of the most fundamental molecular process is called transcription: it is the production of RNAs for a given gene.

Transcription is a quantitative process. Cells can produce varying amounts of RNAs for a given gene depending on the environment or the genotype.



Assessing associations of the strains genotype and their RNA abundance leads to a massive amount of tests.

Install library `genefilter`

We start by installing `genefilter`, an R package which among other things allows fast iteration of testing with the T-test over rows of a matrix. This call will prompt the question "Update all/some/none? [a/s/n]:", answer none (press 'n').

```
#install.packages('BiocManager')
#BiocManager::install("genefilter")
library(genefilter)
```

Gene expression data

The expression data records the gene expression of 8,382 genes, of different strains in different environments. The units are arbitrary.

```
expression <- fread("extdata/eqtl/expression.txt")
# subset to the strains grown in YPMalt.
expression <- expression[ , grep('YPMalt', colnames(expression)), with=FALSE]
# rename columns to only identify the strain.
colnames(expression) <- sapply(strsplit(colnames(expression), '-'), function(x) x[2])

expression[, c(1:8)]
```

```
##          seg_03A    seg_04C    seg_07C    seg_07D     seg_08D     seg_09D
##    1: -1.3726907 -1.8847238 -1.8296967 -1.1234236 -1.82782590 -1.20675410
##    2:  0.5713398  1.2145694 -0.2335415  0.3083605  0.04598717  0.03109929
##    3: -0.1518188  0.7454191 -0.1056484  0.2339785 -0.88028818  0.03693013
##    4:  1.5542958  1.7350163  1.4120586  0.7520013  1.47756501  1.19606233
##    5: -0.3941735 -2.3182249 -2.2795963 -0.4134882 -0.44609369 -0.17299278
##   ---
## 8378: -1.2376779 -0.9263275 -0.4181500 -0.7935673 -1.01801739 -0.37509545
## 8379:  0.4898132  0.9001743  1.2128318  1.7870922  0.06007709  0.83247163
## 8380:  1.7562466  2.0746287  2.1970278  2.4437495  1.43989625  1.53737233
## 8381:  2.6916811  2.7319828  3.2178792  2.9504275  2.36894829  2.73182403
## 8382: -0.0628002  0.7997576  0.0102297 -1.3198039 -0.03482682 -0.49378698
##          seg_10A     seg_10B
##    1: -0.5681886 -2.15923210
##    2:  1.5231203 -0.75517788
##    3: -0.2010784  0.32226112
##    4:  0.9271267  1.39981791
##    5: -0.5496652 -2.09918130
##   ---
## 8378: -0.7367380 -0.07774434
## 8379:  2.2348297  1.21666722
## 8380:  3.1724342  1.43227463
## 8381:  3.3808922  2.46395627
## 8382: -0.5947358 -0.28697953
```

Prepare the genotype matrix

```
# Prepare the genotype matrix for eQTL testing. This time it is better to have it
# in a wide format. So let's just read it in again.

genotype <- fread("extdata/eqtl/genotype.txt")
# subset the genotype matrix to the samples for which expression data exists.
genotype_expression_profiled <- genotype[strain %in% colnames(expression)]
# drop the strain column.
genotype_expression_profiled <- genotype_expression_profiled[,-1]
genotype_expression_profiled[1:5, 1:5]
```

```
##           mrk_1       mrk_14       mrk_27       mrk_40       mrk_54
## 1: Wild isolate Wild isolate Wild isolate Wild isolate Wild isolate
## 2:   Lab strain   Lab strain   Lab strain   Lab strain   Lab strain
## 3:   Lab strain   Lab strain   Lab strain   Lab strain   Lab strain
## 4: Wild isolate Wild isolate Wild isolate Wild isolate Wild isolate
## 5: Wild isolate Wild isolate Wild isolate Wild isolate Wild isolate
```

Gene expression profiling at 8,382 genes (including coding and non-coding RNAs) has been performed for 34 strains grown in the same media. The log-expression levels are stored in the matrix 'expression', the genotype information for these 34 strains are in the 'genotype_expression_profiled' data table.

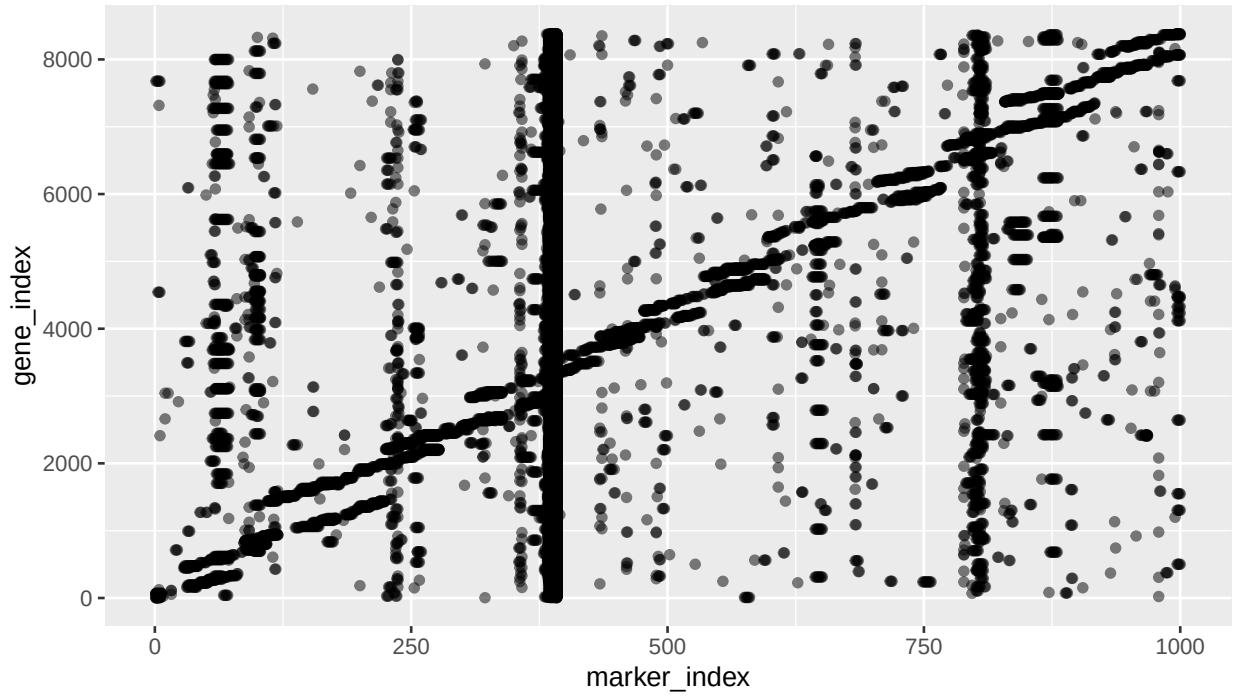
Test and plot the significant associations between genotype and gene expression $FDR < 0.1$.

First, obtain a matrix (# of genes, # of markers) containing all p-values from t-tests between gene expression and genotype. Then, correct it for $FDR < 0.1$ and obtain the significant ones. Scatterplot only the significant associations. Hint: Use the function `rowttests()` from the 'genefilter' package which fastly applies a t-test to each row of a matrix (`as.matrix()`).

```
## for each genotype, perform the QTL mapping to all the expression phenotypes
## you get the matrix of p-values

# t test of gene expression 1 vs marker 1
# t.test(as.numeric(expression[1,]) ~ genotype_expression_profiled[,mrk_1])

# t tests of all gene expressions vs marker 1 using rowttests
# rowttests(as.matrix(expression), fac = factor(genotype_expression_profiled[,mrk_1]))

# t tests of all gene expressions vs all markers
pv = apply(genotype_expression_profiled, 2,
  function(x){
    rowttests(as.matrix(expression), fac = factor(x))$p.value
  }
)

# pv is a matrix containing one p-value for every gene marker combination.
dim(pv)
```

```
## [1] 8382 1000
```

```
## compute the FDR
fdr <- matrix(p.adjust(as.vector(pv), method='BH'),
  ncol = ncol(pv)
)

## get the locations of significant points
signif = which(fdr < 0.1, arr.ind = TRUE)

## eQTL mapping plot
signif <- as.data.table(signif)
colnames(signif) <- c('gene_index', 'marker_index')
ggplot(signif, aes(marker_index, gene_index)) + geom_point(alpha = 0.5)
```




```
## gene expression is mostly regulated locally (diagonal line)
## there are global regulators of gene expression or artifacts (vertical lines)
## the biggest one is at ~390 that affects the overall growth
```

```
dim(pv)
```

```
## [1] 8382 1000
```

OPTIONAL Section 04 - P-values and FDR

Here we will use simulations to investigate the effect of sample size and of the proportion of true and false null hypotheses when performing multiple testing. We will do it for the problem of two-sample comparison with equal sizes.

We are interested in comparing the observations of two samples: $x_1, ..., x_n$ and $y_1, ..., y_n$. Specifically, we ask whether the expectations differ using a two-sample Student t-test.

1. simulate data under the null hypothesis $H_0 : \mu_x = \mu_y$.

We simulate $N = 10,000$ times two samples $x_1, ..., x_n$ and $y_1, ..., y_n$ where X and Y follow the standard normal distribution. We use sample size $n = 50$ but our function works for any `sample_size`. For each simulated dataset, we compute the two-sided p-value of a t-test. We assume unequal variance as by default in the R function `t.test()`.

You can use the following function to do this:

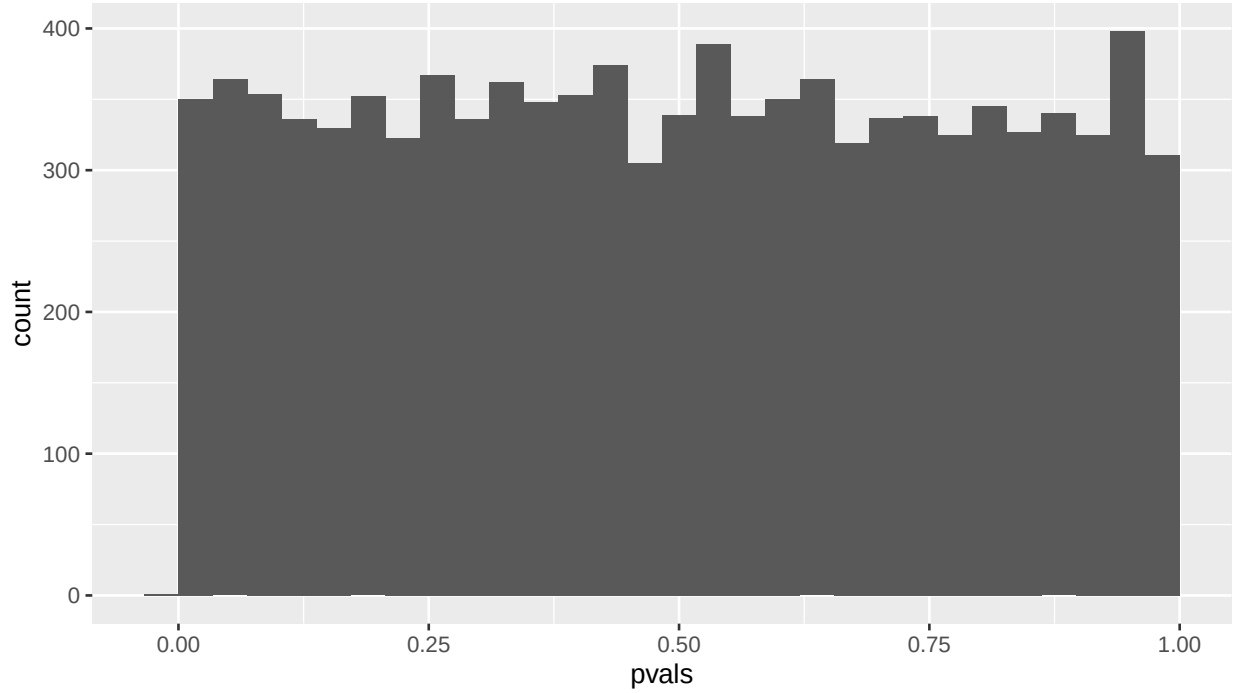
```
simulate_norm_null <- function(sample_size=50, N_experiments=10000){
  sapply(seq(N_experiments), function(i){
    x <- rnorm(sample_size)
    y <- rnorm(sample_size)
```

```
    t.test(x, y, alternative="two.sided")$p.value
  })
}
```

2. Plot the p-values simulated before.

How are the p-values distributed? Create a function that given a vector of p-values, plots a histogram of them. Plot p-values for $sample\_size = 50$. What could be a better visualization?

```
plot_pval <- function(pvals, ...){
  ggplot(data.table(pvals=pvals), aes(pvals)) + geom_histogram(boundary = TRUE)
}

pvals0 <- simulate_norm_null(sample_size=50)
plot_pval(pvals0)
```




```
## Histograms are not always helpful in such a scenario. A Quantile-Quantile plot suits here better.
```

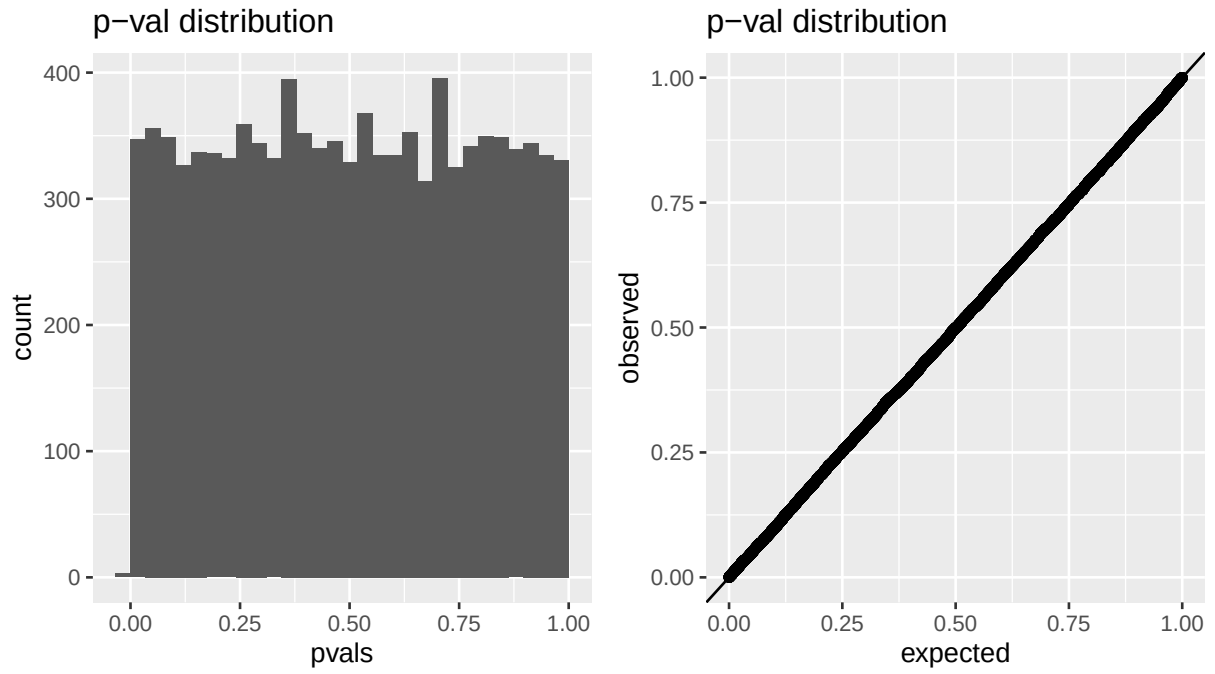
3. compute the quantiles and add a Q-Q plot to the histogram.

If all tests are truely under the null hypothesis, the distribution of the p-values should be uniform by definition. A quantile-quantile plot compares the expected p-values with the observed onse. What are the quantiles for p-values and how can we compute them? Add the Q-Q plot to the function from the last question and add a line where you expect the points to be. Please plot again the p-values for $sample\_size = 50$ with the new function.

Hint: add `title` as a parameter for later.

```
plot_pval <- function(pvals, title="p-val distribution", ...){
```

```
  pval_dt <- data.table(pvals=pvals)
  histo <- ggplot(pval_dt, aes(pvals)) + geom_histogram(boundary = TRUE) +
    labs(title = title)

# Option using only ggplot:
#  qq <- ggplot(data = pval_dt, aes(sample = pvals)) +
#    geom_qq(distribution =  stats::qunif, dparams = list(min = 0, max = 1)) +
#    stat_qq_line(distribution =  stats::qunif, line.p = c(0, 1)) +
#    labs(title = title)

  qq_dt <- data.table(
    observed = sort(pvals),
    expected = ppoints(length(pvals))
  )
  qq <- ggplot(qq_dt) +
    geom_point(aes(expected, observed)) +
    geom_abline(intercept = 0, slope = 1) +
    xlim(0,1) + ylim(0,1) +
    labs(title = title)

  histo + qq

}

pvals0 <- simulate_norm_null(sample_size=50)
plot_pval(pvals0)
```

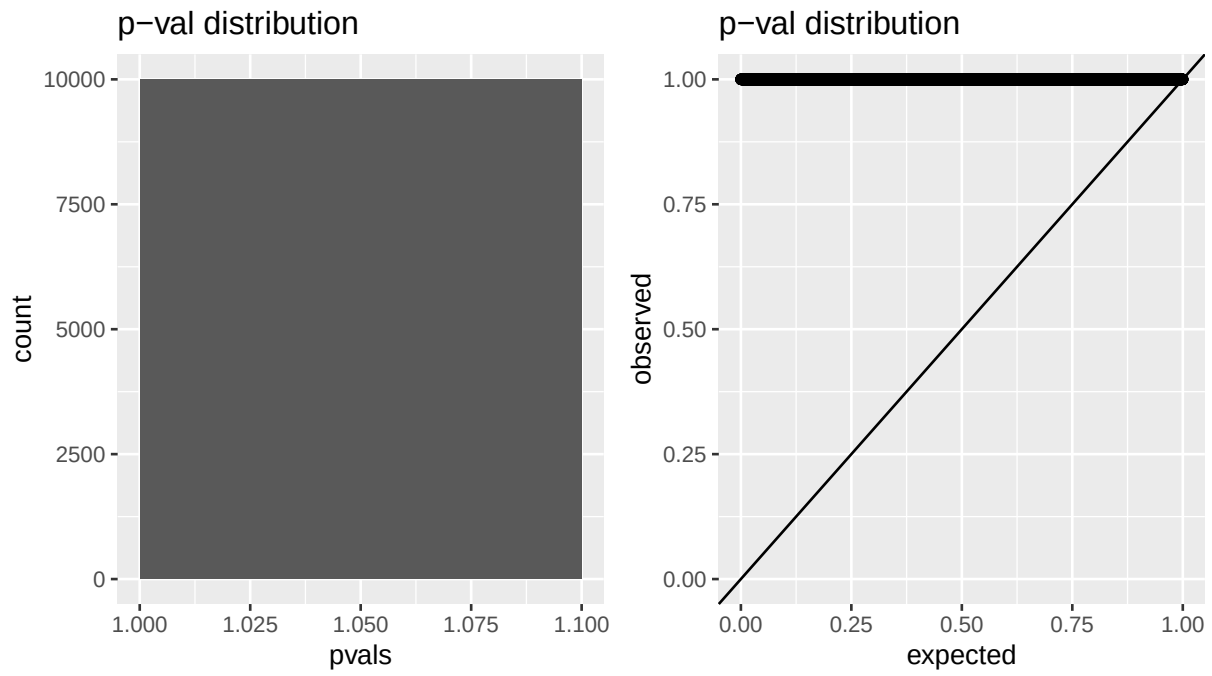



```
## For the uniform distributed p-values we can use
## the ppoint function to generate the expected quantiles.
```

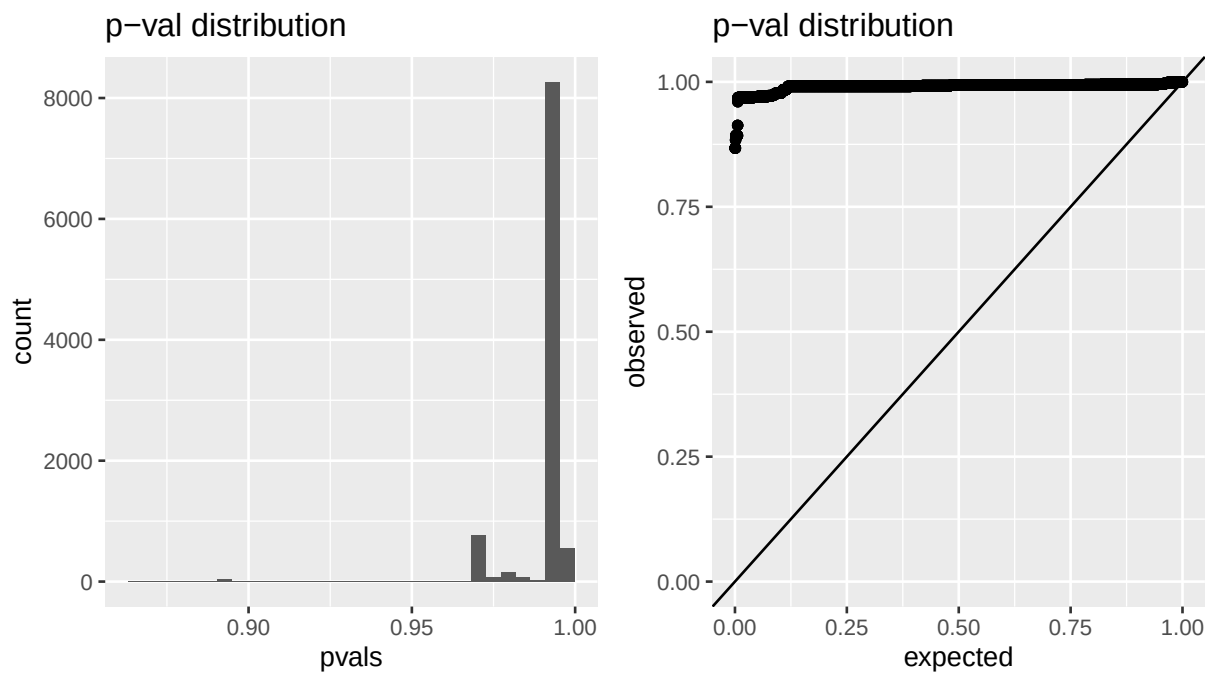
4. Correct for multiple testing

Adjust p-values with the different methods seen in the class. Plot the results using the plot function. Do they behave as expected? Discuss.

```
pvals0_B <- p.adjust(pvals0, m = 'bonferroni')
plot_pval(pvals0_B, main = "\nBonferroni adj p-values")
```




```
pvals0_BH <- p.adjust(pvals0, m = 'BH')
plot_pval(pvals0_BH, main = "\nBenjamini-Hochberg adj p-values")
```

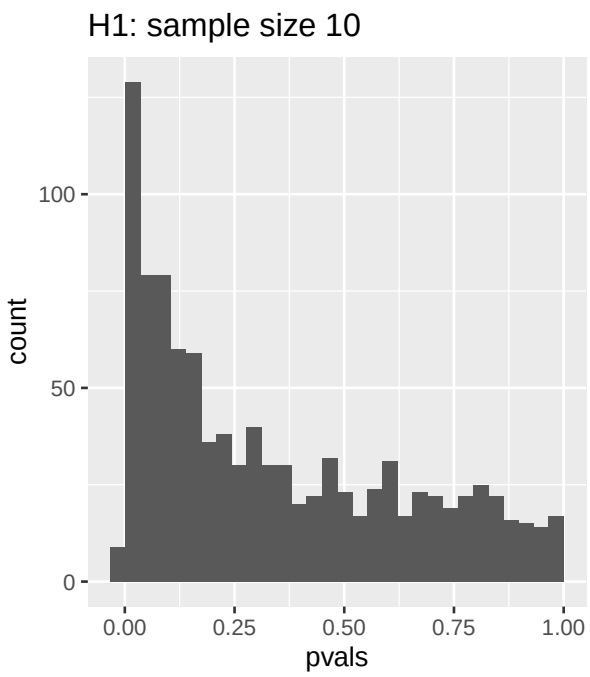


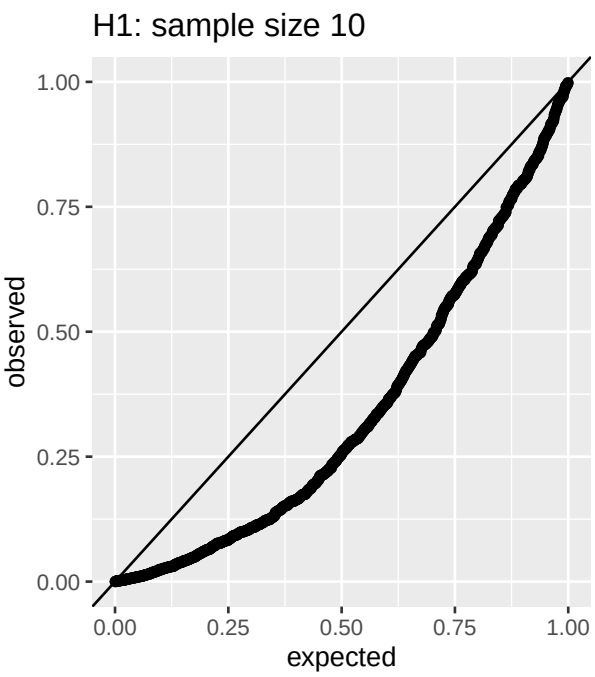

```
## The adjusted p-values do not behave as the nominal p-values.
## They do not come from a uniform distribution and hence
## the histogram as well as the Q-Q plot looks skewed.
## This makes sense: our adjustments account for the fact that p-values
## Have a uniform distribution and corrects them upwards to remove
##  false positives
```

5. Simulate data under the alternative hypothesis $H_1 : \mu_x \neq \mu_y$

Simulate $N = 1,000$ times two samples $x_1, ..., x_n$ and $y_1, ..., y_n$ where X and Y follow the normal distribution with $\mu_x = 0$ and $\mu_y = 0.5$ with a $sample\_size = 10$. For each simulated dataset, compute the two-sided p-value of a t-test. You can assume unequal variance as by default in the R function `t.test()`. Create the same p-value plots as done before.

```
simulate_norm_alt <- function(sample_size = 100, N_experiments = 1000, mu_diff = 0.5){
  sapply(seq(N_experiments), function(i){
    a <- rnorm(sample_size)
    b <- rnorm(sample_size, mu_diff)
    t.test(a, b)$p.value
  })
}

plot_pval(simulate_norm_alt(sample_size=10), title="H1: sample size 10")
```

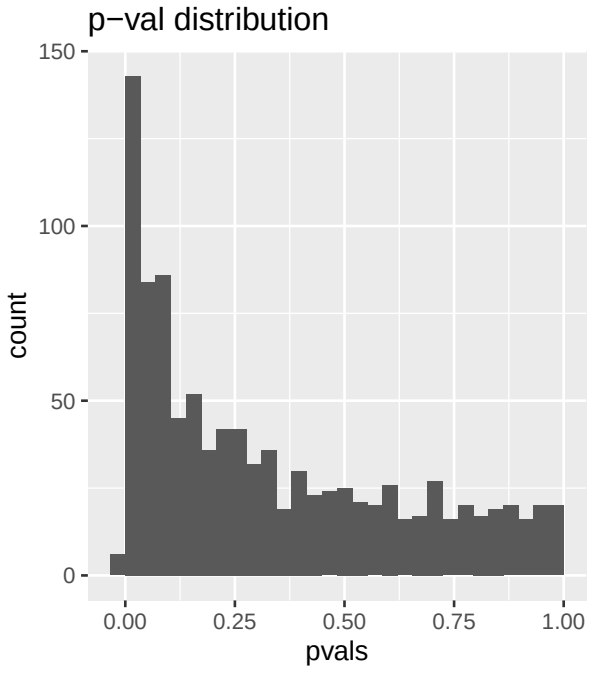


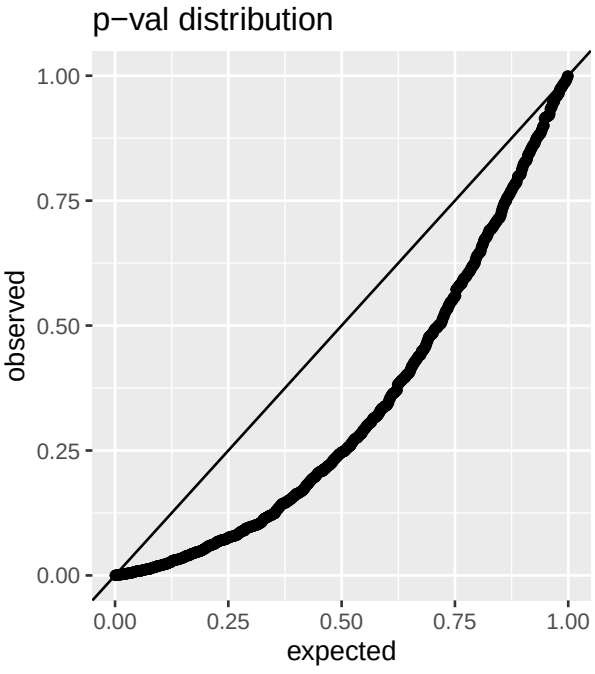


OPTIONAL Section 05 - sample size and power

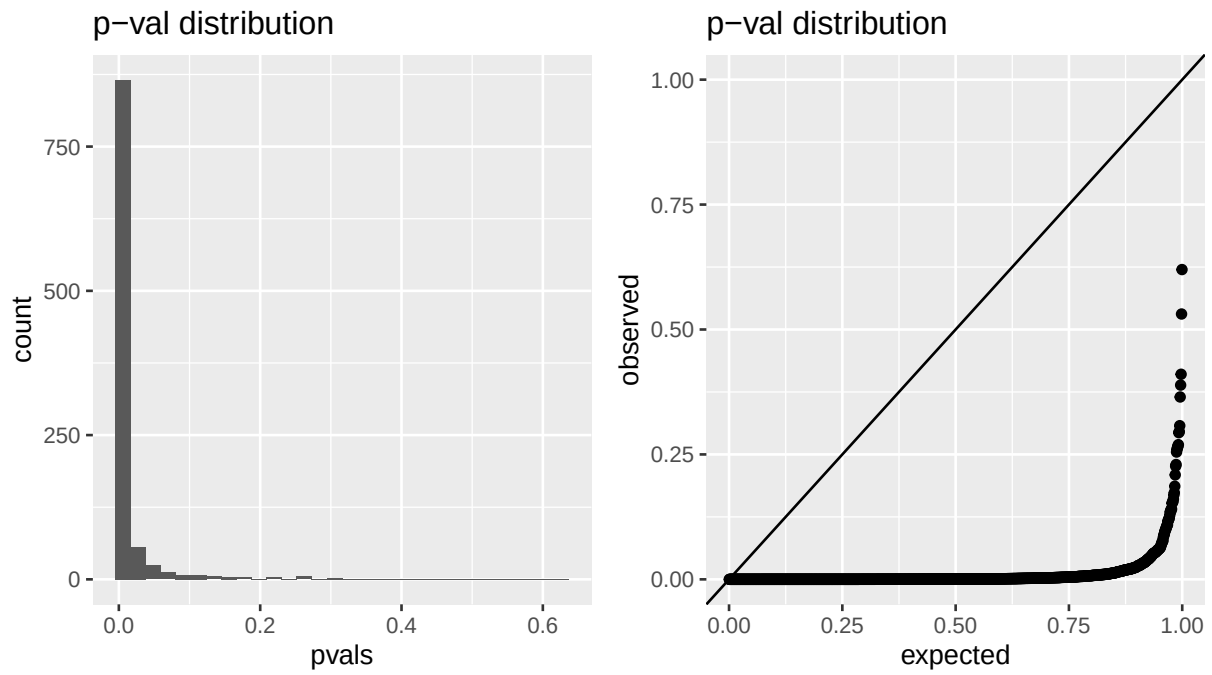
1. Investigate the effect of different sample sizes n (10, 100, 1000) on the p-value plots of the question above. Discuss.

```
plot_pval(simulate_norm_alt(sample_size=10), main="H1: sample size 10")
```

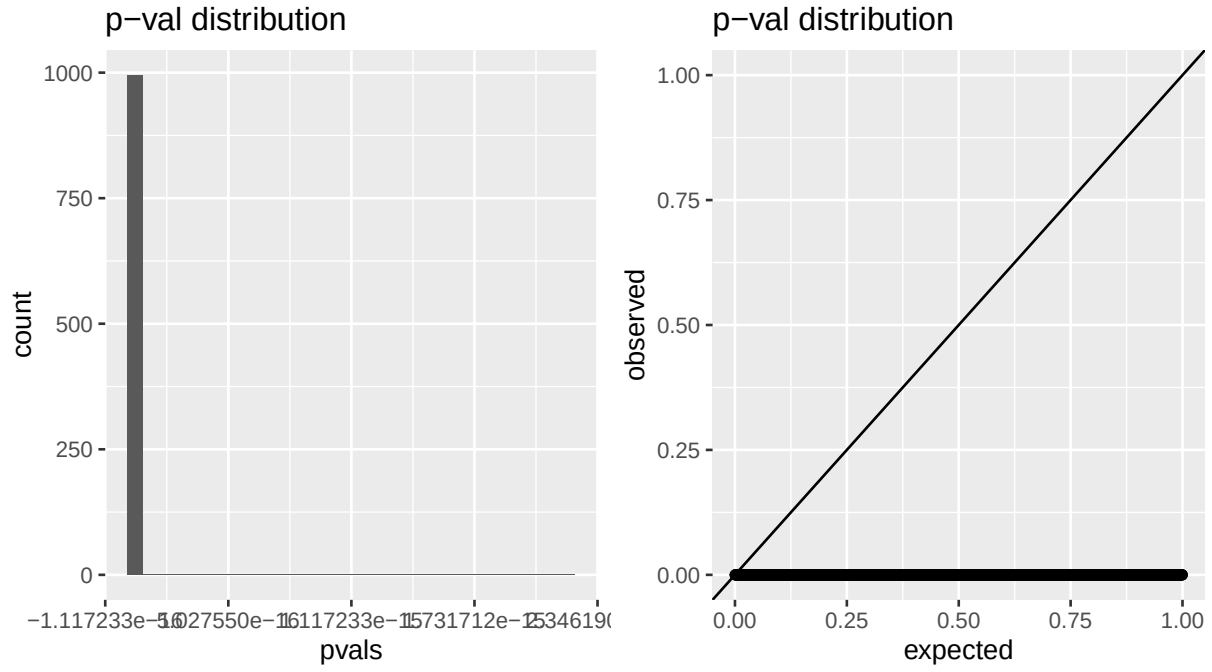




```
plot_pval(simulate_norm_alt(sample_size=100), main="H1: sample size 100")
```




```
plot_pval(simulate_norm_alt(sample_size=1000), main="H1: sample size 1000")
```

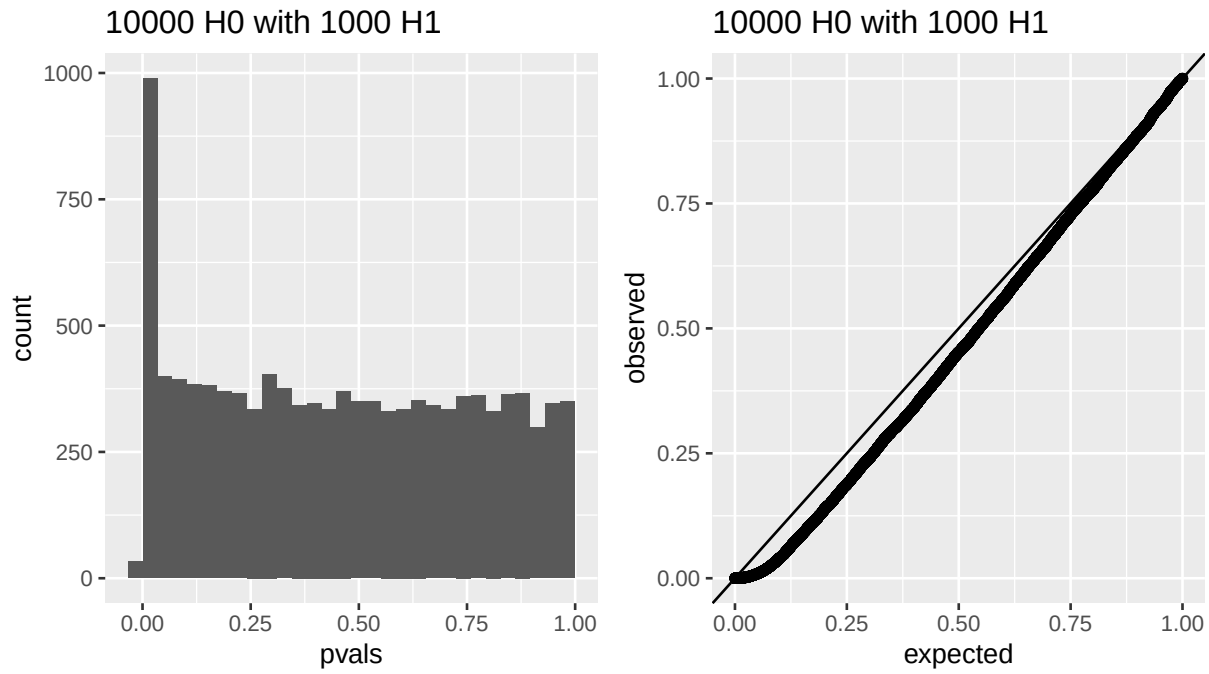


```
## The higher the number of observations the lower the p-value gets.
## This means a tiny differences can be found significant
## if one has enough observations.
```

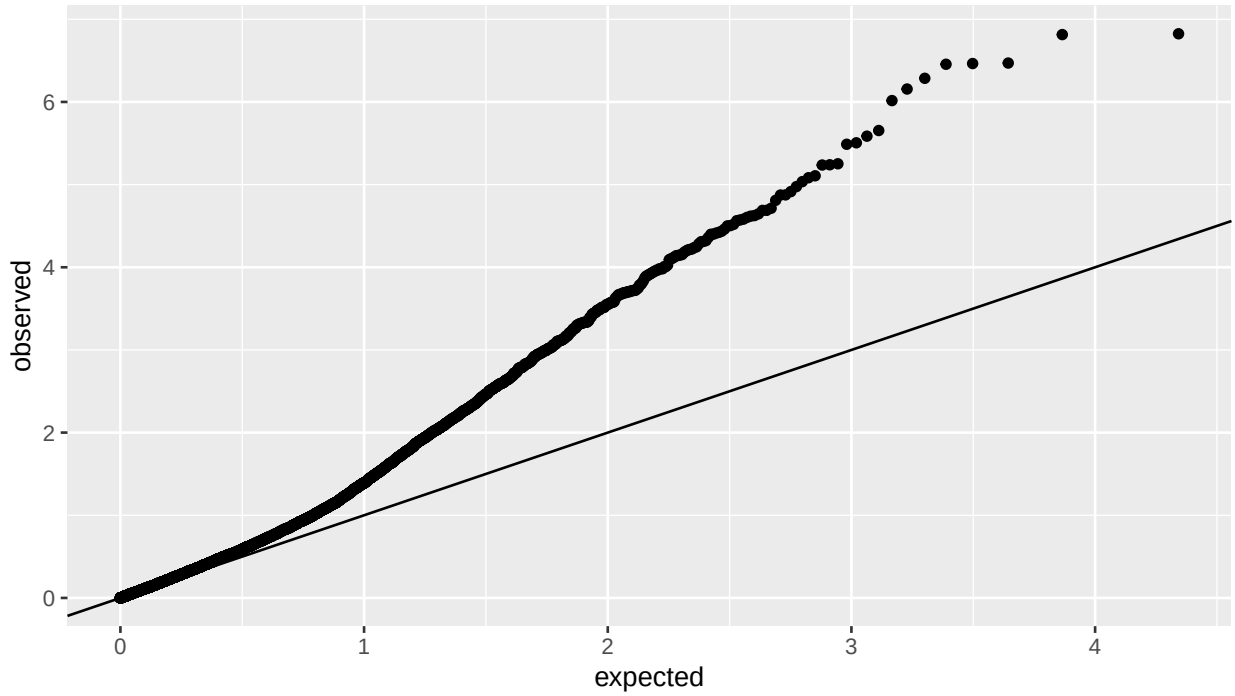
2. p-values for a mixture of null and alternative.

Provide the same plots as before when considering a dataset of $N_0 = 10000$ data points simulated under H_0 (true null) and $N_1 = 1000$ data points simulated under H_1 (false null). Discuss. For p-values one is mostly interested in the lower range. Think of a transformation on how to visualize the p-value data in a better form and change the plotting function accordingly.

```
pvals <- c(
  simulate_norm_null(sample_size = 50, N_experiments = 10000),
  simulate_norm_alt(sample_size = 50, N_experiments = 1000))
plot_pval(pvals, title = "10000 H0 with 1000 H1")
```




```
## A -log10(...) transformation is helpful in the context of p-values.
plot_pval_log10 <- function(pvals, title="p val distribution", ...){
  n  <- length(pvals)
  dt <- data.table(
    observed = -log10(sort(pvals)),
    expected = -log10(ppoints(n))
  )
  ggplot(dt) +
    geom_point(aes(expected, observed)) +
    geom_abline(intercept = 0, slope = 1)

}
plot_pval_log10(pvals, main = "10000 H0 with 1000 H1")
```



3. Mixture of H_0 and H_1 adjusted for multiple testing

Adjust the p-values with Benjamini-Hochberg (FDR) in the mixture from the previous question. Make a contingency table of true positives, true negatives, false positives and false negatives. Try this with different sample sizes for FDR = 0.05. Discuss.

Do the same thing for the bonferroni correction and compare the results

Hint: Create a function with at least `sample_size` as parameter. Use `names(pvals) <- rep(c("H0", "H1"), c(10000, 1000))` to name your p-values. You can then use `table` to create your contingency matrix.

```
error_analysis <- function(method='BH', sample_size=50, cut=0.05){
  pvals <- c(
    simulate_norm_null(sample_size, N_experiments=10000),
    simulate_norm_alt(sample_size, N_experiments=1000))

  names(pvals) <- rep(c("H0", "H1"), c(10000, 1000))

  pvals_adj <- p.adjust(pvals, method=method)
  table(ifelse(pvals_adj < cut, "significant", "not significant"), names(pvals))
}
set.seed(10)
error_analysis(sample_size = 10)
```

```
##
##                      H0    H1
##   not significant 10000  1000
```

```
error_analysis(method="bonferroni", sample_size = 10)
```

```
##
```

```
##                       H0    H1
##   not significant 10000  1000
```

```
error_analysis(sample_size = 30)
```

```
##
##                       H0    H1
##   not significant 10000   979
##   significant         0    21
```

```
error_analysis(method="bonferroni", sample_size = 30)
```

```
##
##                       H0    H1
##   not significant 10000   998
##   significant         0     2
```

```
error_analysis(sample_size = 50)
```

```
##
##                      H0   H1
##   not significant 9991  775
##   significant        9  225
```

```
error_analysis(method="bonferroni", sample_size = 50)
```

```
##
##                       H0    H1
##   not significant 10000   986
##   significant         0    14
```

```
error_analysis(sample_size = 100)
```

```
##
##                      H0   H1
##   not significant 9965  274
##   significant       35  726
```

```
error_analysis(method="bonferroni", sample_size = 100)
```

```
##
##                       H0    H1
##   not significant 10000   862
##   significant         0   138
```

```
error_analysis(sample_size = 1000)
```

```
##
##                      H0   H1
##   not significant 9947    0
##   significant       53 1000
```

```r
error_analysis(method="bonferroni", sample_size = 1000)
```

```
##
##                       H0    H1
##   not significant 10000     0
##   significant         0  1000
```

```r
# We see the trade-off:
# The bonferroni produces less false positives
# But at the cost of considerably more false negatives
# unless the sample size is huge
```